

중간보고서

분과명: D(지능형융합보안)

과제명: 블록체인 기반의 신재생에너지 거래 플랫폼 개발

팀명: 아메리카노 샷추가

201724595 차민준

201724436 김재현

201724508 오재석

담당 교수: 최윤희 교수님

<목차>

1. 요구 조건 및 제약 사항 분석	2
1.1. 과제 목표	2
1.2. 요구조건	2
2. 설계 상세화	4
2.1. 네트워크 구성/변경 예정 사항	4
2.2. 체인코드 및 채널	5
2.3. Front-end	6
2.4. Back-end	7
3. 과제 추진 계획	8
3.1. 현재 과제 계획	8
3.2. 향후 과제 계획	8
4. 과제 진행 내용	9
4.1. 구성원별 진행중 작업	9
4.2. 보고 시점까지의 과제 수행 내용 및 중간 결과	10

1. 요구 조건 및 제약 사항 분석

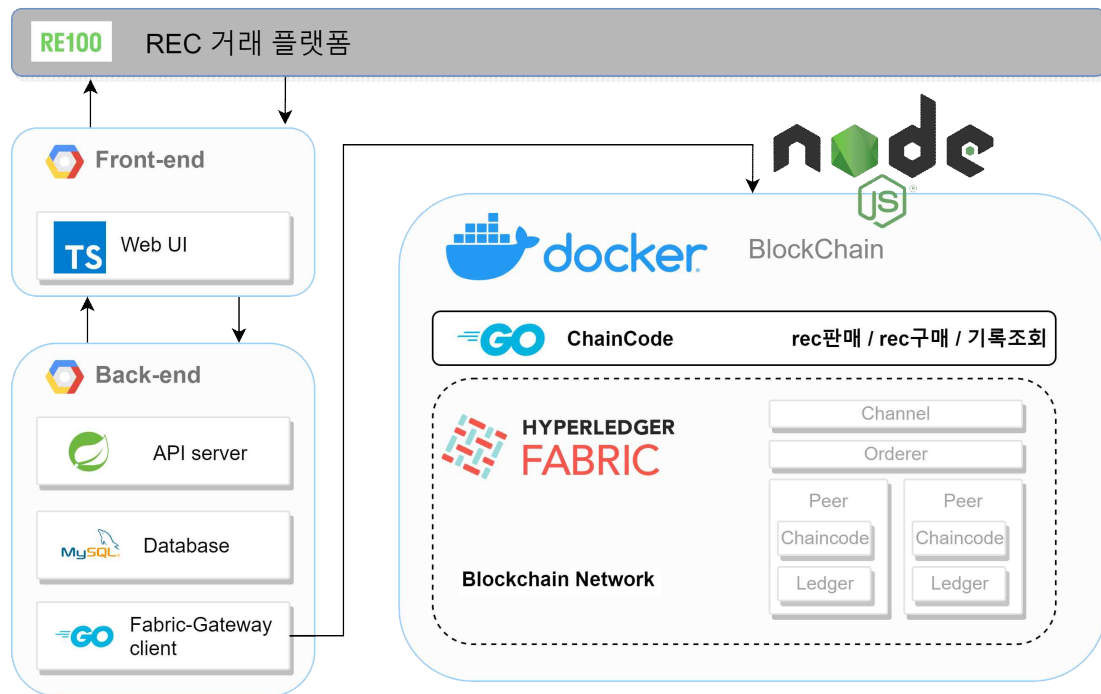


그림 1 [과제 전체 구상도]

1.1. 과제 목표

- 1.1.1. 블록체인을 기반으로 하여 re100이행을 위한 REC코인을 안전하게 거래할 수 있는 플랫폼 개발
- 1.1.2. 하이퍼래저 패브릭을 사용하여 체인코드를 통해 REC코인 판매, 구매 기능을 제공
- 1.1.3. 사용자의 역할(판매자, 구매자)에 따라 서로 다른 기능 제공

1.2. 요구조건

1.2.1. 블록체인 네트워크

조직(Org)	• 판매자 / 구매자 / 국세청 / 전력거래소 4가지 종류로 구성
채널	• 1개의 채널 내에서 조직마다 다른 권한을 부여/ 채널을 분리 두 가지 방법 중 한 가지 선택

1.2.2. Front-end 및 Back-end

Front-end	<ul style="list-style-type: none"> • Html 페이지로 작성된 Front-end • 직관적인 UI를 사용하여 사용자 친화적인 서비스 제공
Back-end	<ul style="list-style-type: none"> • spring boot로 (혹은 nodejs)된 back-end • 설치할 chaincode에 내장된 Fabric-Gateway client를 이용해 도커 컨테이너들과 통신 • 사용자를 특정해 줄 로그인 서비스 제공

1.2.3. 웹 서비스

- 사용자 접근 편의성 제공을 위한 웹 서비스
- 사용자별 ID/PW/블록체인 권한 부여

국세청	거래 발생 시 세금 징수
전력거래소	모든 거래 열람 가능 / 블록체인 특성에 따라서 거래 수정 불가
판매자	REC코인 판매 가능
구매자	REC코인 구매 가능

- ChainCode설치 시 policy를 달리하여 권한 부여

1.2.4. 제약사항 분석 및 수정사항

블록체인 네트워크	제약사항	실제로는 사용자의 수 만큼 조직(Org)가 필요
	해결방안	현재 프로그램에서는 두 개의 조직이 있는 것으로 가정
REC 등록	제약사항	REC등록의 경우 검증된 기관에서 오프라인상으로 설비 확인 후 생산된 전력에 대해서만 REC부여
	해결방안	검증을 생략하고 asset 생성 시 판매자가 직접 등록할 수 있도록 진행
로그인	제약사항	신재생 에너지 거래 플랫폼에서 사용자를 구분할 로그인 시스템 구축 필요
	해결방안	mysql을 사용한 로그인 서비스와 블록체인 네트워크의 연계 계획 중

2. 설계 상세화

2.1. 네트워크 구성/변경 예정 사항

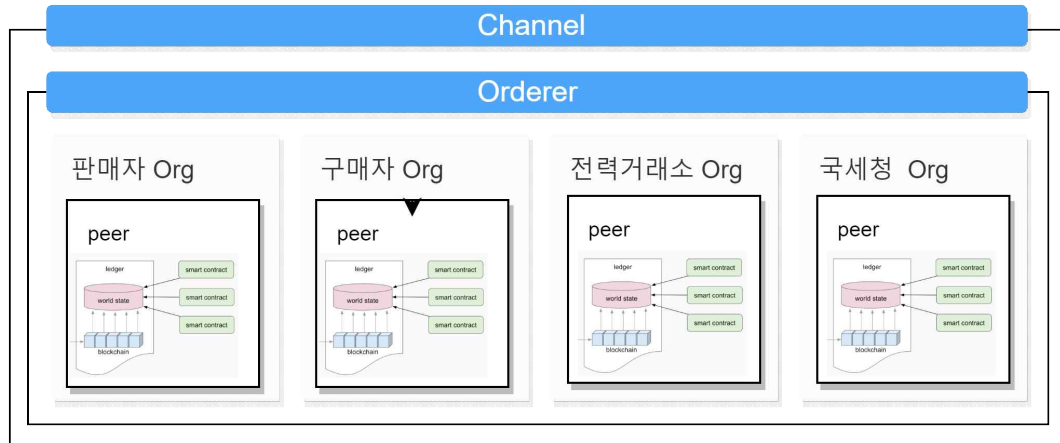


그림 2 현재 블록체인 네트워크 구성

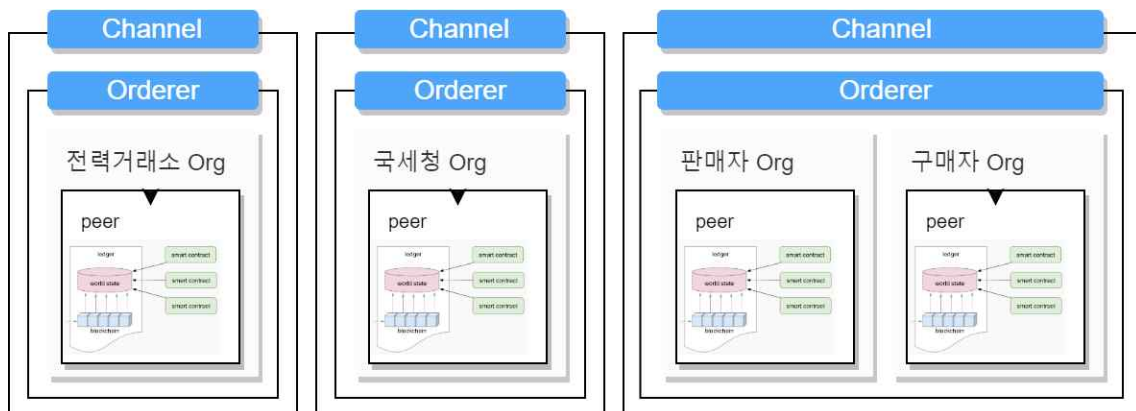


그림 3 변경할 블록체인 네트워크 구성

현재 진행	<p>한 개의 채널/orderer 사용</p> <p>판매자/구매자/전력거래소/국세청 조직 사용</p> <p>회원가입시 판매자/구매자 조직 추가</p> <p>조직별 권한 부여</p>
변경 예정 사항	<p>판매자/구매자/전력거래소/국세청 조직들을</p> <p>판매자/구매자 채널, 전력거래소 채널, 국세청 채널 분리</p> <p>각 채널별 orderer사용 및 Raft 합의 알고리즘 사용</p>

2.2. 체인코드 및 채널

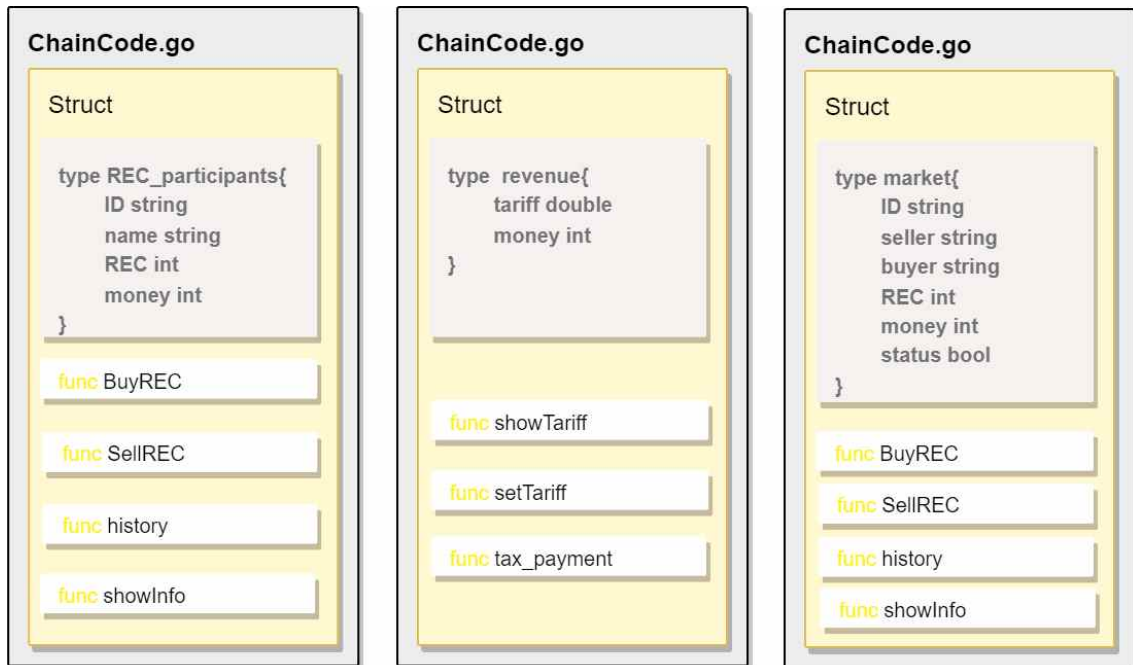


그림 4 채널별 구조체와 체인코드

현재	하나의 채널에 모든 조직을 구현 후 그 조직끼리 거래를 진행.(블록체인 내부데이터가 아닌 mysql에 권한을 부여하여 실행)	
변경 예정	채널을 성격에 따라 3가지 채널로 분할	
	국세청	REC코인 세율 / 들어온 세금 저장 세율 조정 / 조회 기능 세금 납부 기능 위 기능이 가능하도록 smart contract 구현
	전력거래소	현재 거래 열람 거래 등록 거래 완료 거래 기록 열람 위 기능이 가능하도록 smart contract 구현
	판매자 구매자	REC코인 판매 가능 REC코인 구매 가능 위 기능이 가능하도록 smart contract 구현

2.3. Front-end

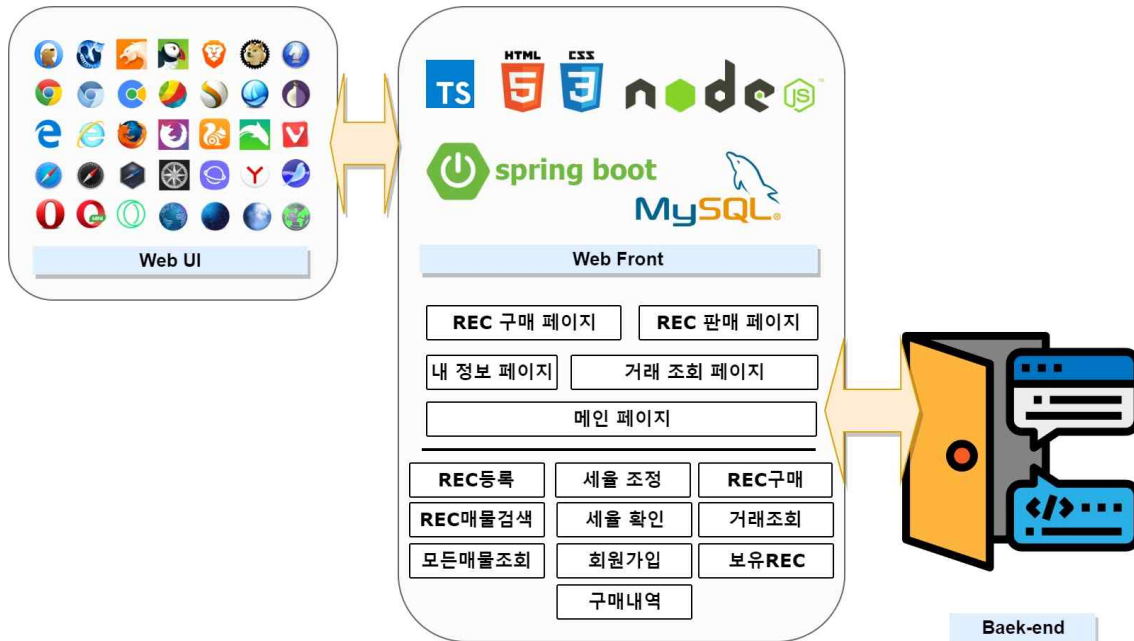
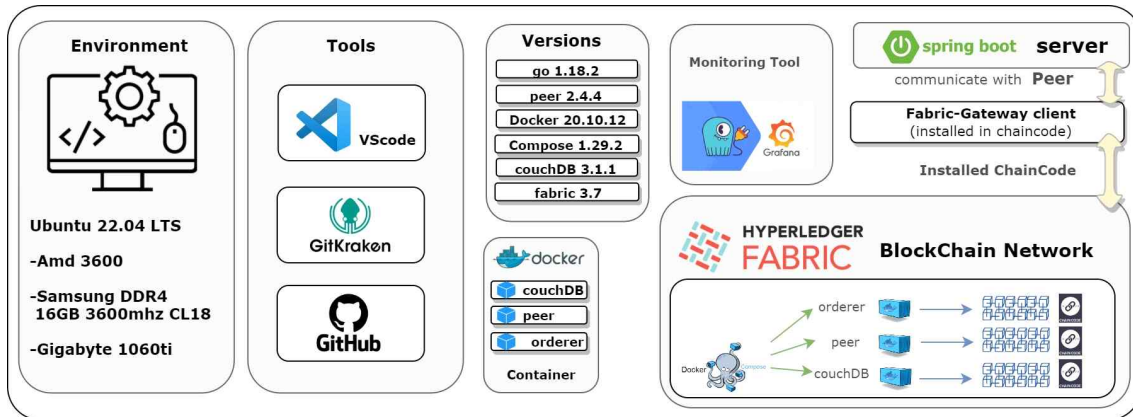


그림 5 Front-end 구조

2.3.1. 웹UI를 제작하여 REC거래를 편하게 할 수 있도록 함
각 권한은 Invoke, Query Policy를 설정해준다.

권한별 UI/UX			
관리자	모든 권한 (Grafana관리 툴 제공)		
	국세청	세울 조정 가능	
	전력거래소	모든 거래 조회 모든 거래 기록 조회	
		사용자	세울 확인 본인의 거래 조회 본인의 REC조회 REC매물 등록/구매 매물 검색 / 조회 본인의 자산 조회//

2.4. Back-end



2.4.1. 목표

서버	<ul style="list-style-type: none"> • spring-boot서버를 통해 fabric-network와 통신 • Grafana를 사용해서 블록체인 네트워크의 상태 점검
데이터베이스	<ul style="list-style-type: none"> • MySQL을 사용해 사용자 정보 저장 • couchDB를 사용해 각 조직 피어의 원장에 world/private 상태 저장
Fabric Gateway	<ul style="list-style-type: none"> • chaincode에 go언어로 작성된 fabric-gateway, binary로 작성된 peer를 통해 블록체인 네트워크와 통신

3. 과제 추진 계획

3.1. 현재 과제 계획

[illegible]

3.2. 향후 과제 계획

[illegible]

4. 과제 진행 내용

4.1. 구성원별 진행중 작업

차민준	블록체인 네트워크 개발 <ul style="list-style-type: none">• 사용자/국세청/전력거래소 ChainCode 개발• Test-network 개발• Test-frontend 개발
김재현	블록체인 네트워크 구축 <ul style="list-style-type: none">• 네트워크 채널 정의/조직 정의• docker/docker-compose 및 조직 config 구성 및 작성• 네트워크 빌드 쉘 스크립트 작성
오재석	플랫폼 개발 <ul style="list-style-type: none">• spring-boot 서버 개발• mySQL 데이터베이스 구축• 로그인/로그아웃 디자인

4.2. 보고 시점까지의 과제 수행 내용 및 중간 결과

1) 블록체인 네트워크

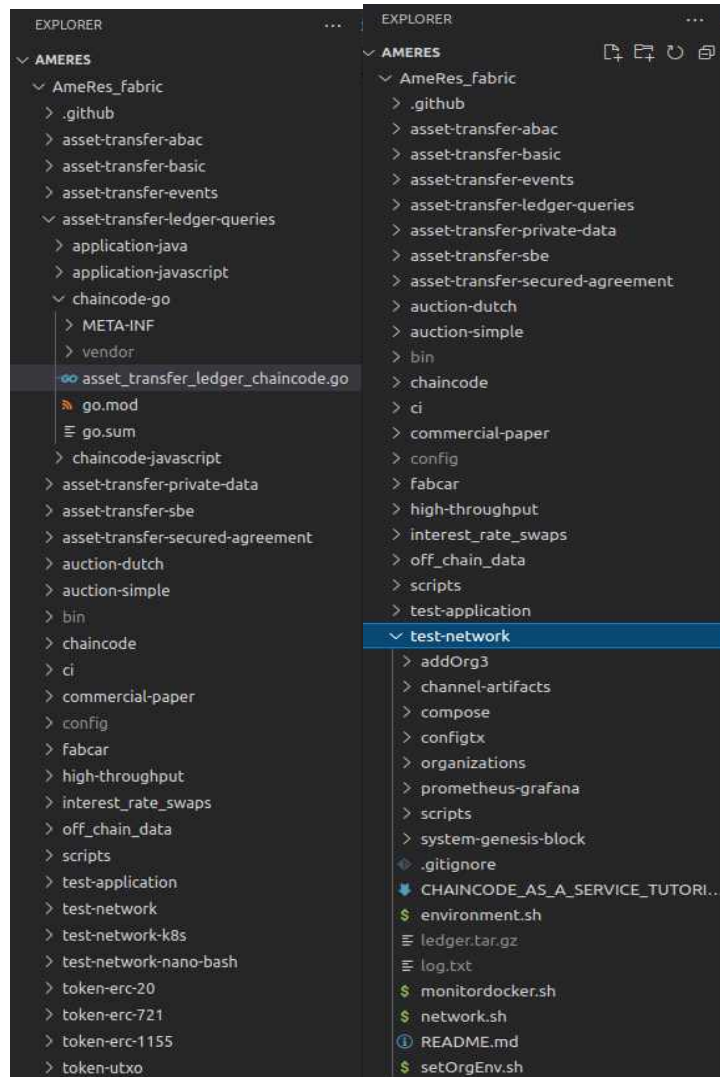
hyperledger fabric의 공식 문서를 통해서 튜토리얼을 진행했다. 그리고 각 용어에 대한 정의, hyperledger fabric이 어떤 구조로 이루어져 있는지 자세하게 적혀있어서 참고하였다.

```
cha@cha-B450M-AORUS-ELITE: ~/fabric_example$ curl -sSL https://bit.ly/2ysb0FE | bash -s

Clone hyperledger/fabric-samples repo

====> Cloning hyperledger/fabric-samples repo
'fabric-samples'에 복제합니다...
remote: Enumerating objects: 10607, done.
remote: Counting objects: 100% (348/348), done.
remote: Compressing objects: 100% (240/240), done.
remote: Total 10607 (delta 133), reused 231 (delta 85), pack-reused 10259
오브젝트를 받는 중: 100% (10607/10607), 19.21 MiB | 12.21 MiB/s, 완료.
델타를 알아내는 중: 100% (5660/5660), 완료.
fabric-samples v2.4.4 does not exist, defaulting to main. fabric-samples main branch is intended to work with recent versions of fabric.
```

[그림] hyperledger fabric clone



[그림] blockchain directory 구조

chaincode-go 디렉토리 내부의 smartcontract 파일을 수정해서 본 팀에서 원하는 asset을 만들 수 있고 함수를 추가할 수 있다.

```
cha@cha-B450H-AORUS-ELITE:~/fabric_example/fabric-samples/test-network$ ./network.sh createChannel
Using docker and docker-compose
creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
Bringing up network
LOCAL_VERSION=2.4.4
DOCKER_IMAGE_VERSION=2.4.4
Starting peer0.org1.example.com ... done
Starting orderer.example.com ... done
Starting peer0.org2.example.com ... done
Starting cli ... done
```

[그림] 채널 생성

```
cha@cha-B450H-AORUS-ELITE:~/fabric_example/fabric-samples/test-network$ ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/ch
aincode-go -ccl go
Using docker and docker-compose
Deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
Rendering Go dependencies at ../asset-transfer-basic/chaincode-go
```

[그림] chaincode 배포

```
./network.sh deployCC -ccn ledger -ccp ../asset-transfer-ledger-queries/chaincode-go/ -ccl go -ccep "OR('Org1MSP.peer','Org2MSP.peer')"
```

실제 체인코드 배포는 위 코드를 사용해서 배포했다.

https://hyperledger-fabric.readthedocs.io/en/release-2.4/deploy_chaincode.html를 참고해서 직접 체인코드를 blockchain network에 배포하는 작업도 해 보았다.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
34968774be85	dev-peer0.org2.example.com-ledger_1.0-3d8967163803c1f1fed63f90f857c7a32402c21cca15ef2a831f874dcda92f	"chaincode -peer add-"	11 seconds ago	Up 10 seconds
613b3d9e7e51	dev-peer0.org1.example.com-ledger_1.0-3d8967163803c1f1fed63f90f857c7a32402c21cca15ef2a831f874dcda92f	"chaincode -peer add-"	11 seconds ago	Up 10 seconds
b1470996f963	hyperledger/fabric-tools:latest	"/bin/bash"	About an hour ago	Up 11 seconds
e74ac1f92f19	hyperledger/fabric-peer:latest	"peer node start"	About an hour ago	Up 11 seconds
/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp	peer0.org1.example.com	"peer node start"	About an hour ago	Up 11 seconds
654494506880	hyperledger/fabric-peer:latest	"tini -- /docker-ent-"	About an hour ago	Up 12 seconds
/tcp, 7053/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp	peer0.org2.example.com	"orderer"	About an hour ago	Up 12 seconds
4/tcp, :::5984->5984/tcp	couchdb0	"tini -- /docker-ent-"	About an hour ago	Up 12 seconds
588c58e3791	hyperledger/fabric-orderer:latest	"orderer"	About an hour ago	Up 12 seconds
/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp	orderer.example.com	"tini -- /docker-ent-"	About an hour ago	Up 12 seconds
1645314059f1	couchdb:3.1.1	"tini -- /docker-ent-"	About an hour ago	Up 12 seconds
4/tcp, :::7884->5984/tcp	couchdb1			

[그림] network실행 및 chaincode 배포 후 docker의 상태

```
JS index.js U    homepage.html U    asset_tr
fabric-samples > asset-transfer-ledger-queries > chaincode-
89
90 type Asset struct {
91     DocType    string `json:"docType"`
92     ID         string `json:"ID"`
93     Owner      string `json:"owner"`
94     Kw         int    `json:"kw"`
95     KRW        int    `json:"KRW"`
96     Role       string `json:"role"`
97 }
98
```

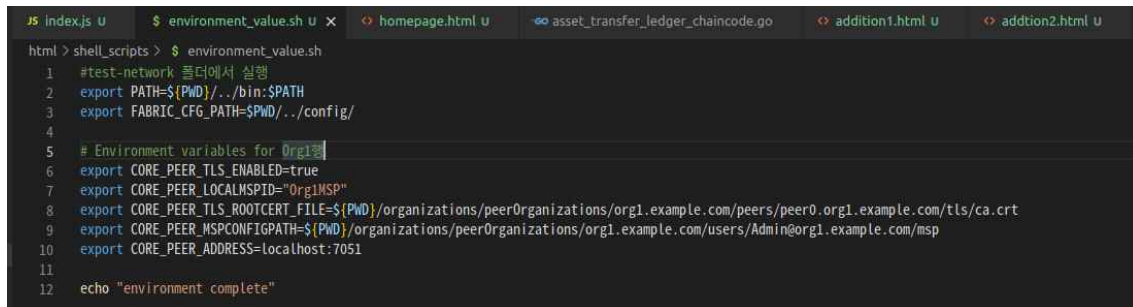
[그림] data structure

우선, smartcontract를 수정해서 본 팀에서 원하는 asset을 만들었다. asset은 개별 사용자를 알 수 있는 ID, 사용자의 이름 Owner, 가지고 있는 전력 Kw, 가지고 있는 재산 KRW, 사이트 내에서의 역할을 나타내는 Role을 내부에 가지고 있다. 본 팀은 role에 index를 적용해서 더 빠른 query가 가능하도록 설계할 예정이다.

```
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$ peer chaincode query -C mychannel -n ledger -c '{"Args":["GetAllAssets"]}'
[{"docType":"asset","ID":"asset5","owner":"tom","kw":40,"KRW":30,"Role":"seller"}]
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n ledger -c '{"Args":["CreateAsset","asset3","tom","35","40","byuer"]}'
2022-07-24 17:25:38.242 KST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$ peer chaincode query -C mychannel -n ledger -c '{"Args":["GetAllAssets"]}'
[{"docType":"asset","ID":"asset3","owner":"tom","kw":35,"KRW":40,"Role":"byuer"}, {"docType":"asset","ID":"asset5","owner":"tom","kw":40,"KRW":30,"Role":"seller"}]
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n ledger -c '{"Args":["DeleteAsset","asset5"]}'
2022-07-24 17:30:03.327 KST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$ peer chaincode query -C mychannel -n ledger -c '{"Args":["GetAllAssets"]}'
[{"docType":"asset","ID":"asset3","owner":"tom","kw":35,"KRW":40,"Role":"byuer"}]
cha@cha-B450M-AORUS-ELITE:~/바탕화면/AmeRes/AmeRes/AmeRes_fabric/test-network$
```

[그림] chaincode 배포 후 invoke

수정한 asset을 적용한 후 chaincode 배포 후 invoke를 해봄. asset을 추가하고, 모든 asset을 보여주고, asset을 delete하는 것이 잘 작동하는 것을 볼 수 있다.

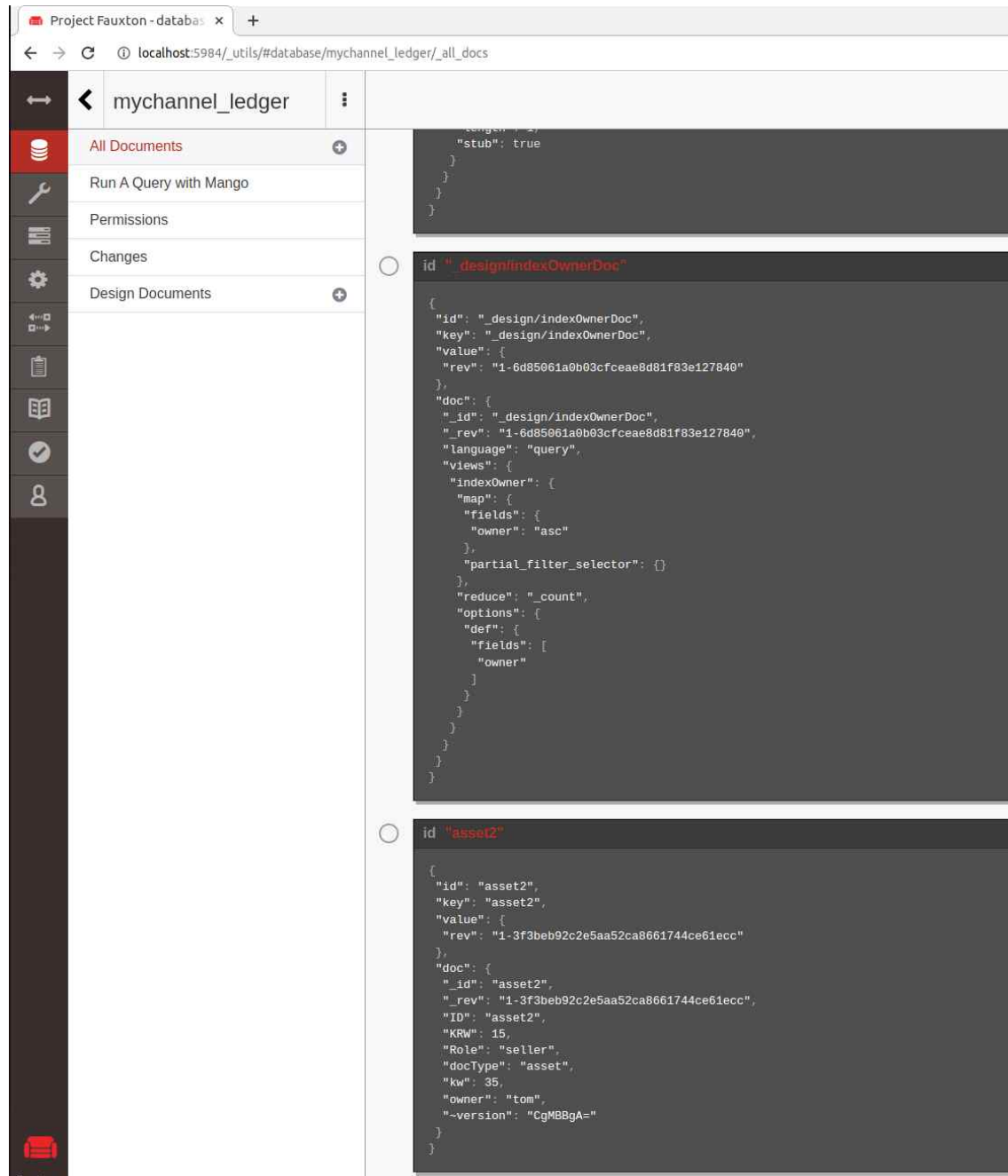


```
html > shell_scripts > $ environment_value.sh
1  #test-network 폴더에서 실행
2  export PATH=${PWD}/../bin:$PATH
3  export FABRIC_CFG_PATH=${PWD}/../config/
4
5  # Environment variables for Org1
6  export CORE_PEER_TLS_ENABLED=true
7  export CORE_PEER_LOCALMSPID="Org1MSP"
8  export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
9  export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
10 export CORE_PEER_ADDRESS=localhost:7051
11
12 echo "environment complete"
```

[그림] shell script

현재는 terminal을 통해서 fabric network의 api를 사용하지만 차후에는 shell script를 통해서 환경 변수 설정과 fabric api를 실행할 수 있도록 만들 계획이다.

그리고 본 팀에서 couchDB를 사용하기로 계획했었다. couchDB와 index를 같이 사용함으로써 특정 조건에서 query문을 상당히 빠르게 실행할 수 있다. 네트워크 생성 시 couchDB를 사용하도록 하였다.



[그림] couchDB

실제 couchDB를 적용하고 이에 접근해 본 모습이다.

본 팀에서 개발하려는 신재생에너지 거래 플랫폼은 궁극적으로 유저 간의 asset거래가 가능해야 한다. 그래서 private data를 도입할 예정이다.

```
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ export PATH=${PWD}/../bin:${PWD}:$PATH
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ export FABRIC_CFG_PATH=${PWD}/../config
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ export FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org1.example.com/
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ fabric-ca-client register --caname ca-org1 --id.name owner --id.secret ownerpw --id.type client --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2022/07/24 21:45:04 [INFO] Configuration file location: /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2022/07/24 21:45:04 [INFO] TLS Enabled
Password: ownerpw
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ fabric-ca-client enroll -u https://owner:ownerpw@localhost:7054 --caname ca-org1 -H "${PWD}/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2022/07/24 21:45:07 [INFO] TLS Enabled
2022/07/24 21:45:07 [INFO] generating key: A[A.ecdsa S:256]
2022/07/24 21:45:07 [INFO] encoded CSR
2022/07/24 21:45:07 [INFO] Stored client certificate at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp/signcerts/cert.pem
2022/07/24 21:45:07 [INFO] Stored root CA certificate at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2022/07/24 21:45:07 [INFO] Stored Issuer public key at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp/IssuerPublicKey
2022/07/24 21:45:07 [INFO] Stored Issuer revocation public key at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp/IssuerRevocationPublicKey
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/owner@org1.example.com/msp/config.yaml"
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ export FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org2.example.com/
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ fabric-ca-client register --caname ca-org2 --id.name buyer --id.secret buyerpw --id.type client --tls.certfiles "${PWD}/organizations/fabric-ca/org2/tls-cert.pem"
2022/07/24 21:45:28 [INFO] Configuration file location: /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org2.example.com/fabric-ca-client-config.yaml
2022/07/24 21:45:28 [INFO] TLS Enabled
Password: buyerpw
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ fabric-ca-client enroll -u https://buyer:buyerpw@localhost:8054 --caname ca-org2 -H "${PWD}/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org2/tls-cert.pem"
2022/07/24 21:45:31 [INFO] TLS Enabled
2022/07/24 21:45:31 [INFO] generating key: A[A.ecdsa S:256]
2022/07/24 21:45:31 [INFO] encoded CSR
2022/07/24 21:45:31 [INFO] Stored client certificate at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp/signcerts/cert.pem
2022/07/24 21:45:31 [INFO] Stored root CA certificate at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp/cacerts/localhost-8054-ca-org2.pem
2022/07/24 21:45:31 [INFO] Stored Issuer public key at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp/IssuerPublicKey
2022/07/24 21:45:31 [INFO] Stored Issuer revocation public key at /home/cha/바탕화면/Anches/Anches/Anches_fabric/test-network/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp/IssuerRevocationPublicKey
chaqcha-8450M-AN00S-ELITE-~/바탕화면/Anches/Anches/Anches_fabric/test-network$ cp "${PWD}/organizations/peerOrganizations/org2.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org2.example.com/users/buyer@org2.example.com/msp/config.yaml"
```

[그림] private data network 설정

private data를 사용하는 모습이다. collections_config.json 파일을 수정해서 정책을 바꿀 수 있다. 현재 네트워크에는 org1, org2가 있는데 서로의 데이터를 읽을 수 없게 설정하였다.

```
// collections_config.json

[
  {
    "name": "assetCollection",
    "policy": "OR('Org1MSP.member', 'Org2MSP.member')",
    "requiredPeerCount": 1,
    "maxPeerCount": 1,
    "blockToLive": 1000000,
    "memberOnlyRead": true,
    "memberOnlyWrite": true
  },
  {
    "name": "Org1MSPPrivateCollection",
    "policy": "OR('Org1MSP.member')",
    "requiredPeerCount": 0,
    "maxPeerCount": 1,
    "blockToLive": 3,
    "memberOnlyRead": true,
    "memberOnlyWrite": false,
    "endorsementPolicy": {
      "signaturePolicy": "OR('Org1MSP.member')",
    }
  },
  {
    "name": "Org2MSPPrivateCollection",
    "policy": "OR('Org2MSP.member')",
    "requiredPeerCount": 0,
    "maxPeerCount": 1,
    "blockToLive": 3,
    "memberOnlyRead": true,
    "memberOnlyWrite": false,
    "endorsementPolicy": {
      "signaturePolicy": "OR('Org2MSP.member')",
    }
  }
]
```

[그림] collections_config.json


```

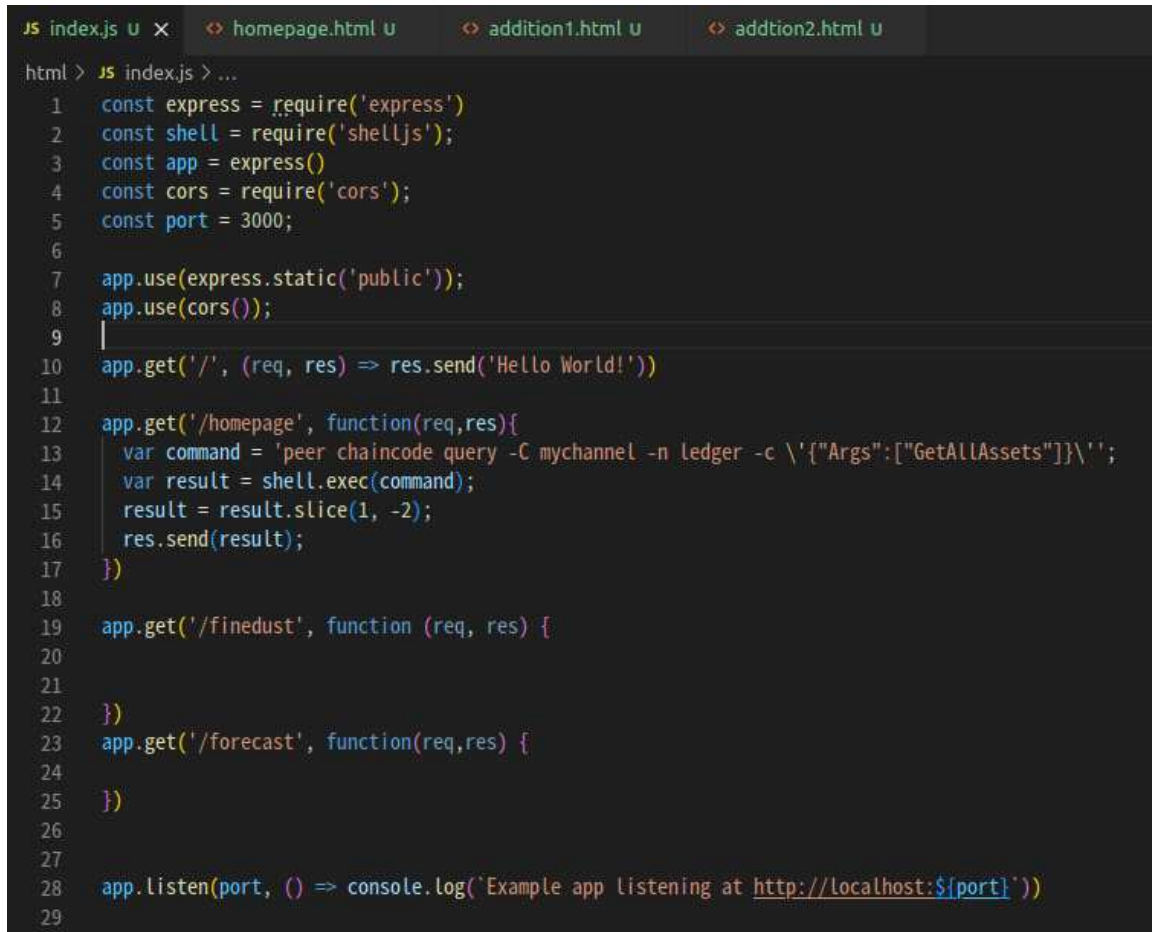
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export PATH=$PWD/.:/bin:/usr/bin
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export FABRIC_CFG_PATH=$PWD/.:/config/
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_TLS_ENABLED=true
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_LOCALMSPID="Org1MSP"
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_ADDRESS=localhost:7051
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_ADDRESS=localhost:7051
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export ASSET_PROPERTIES=${echo -n '{"objectType":"asset","asset":{"color":"green"},"size":20,"owner":"509:Ch-owner,0b-client,0-buyer,ledger-ST-North Carolina,C405:Ch-ca.org1.example.com,0-org1.example.com,L-Durham,ST-North Carolina,C405"}' | base64 | tr -d '\n'}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode invoke -n localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"CreateSet","Args":[""]}' --transient '{"asset:properties","$ASSET_PROPERTIES"}'
2022-07-24 21:46:03.201 IST 0001 INFO [chaincodeCmd] chaincodeInvokeStubQuery -> Chaincode invoke successful, result: status:200
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -c mychannel -n private -c '{"function":"ReadAsset","Args":["asset1"]}'
{"objectType":"asset","asset":{"assetID":"asset1","color":"green","size":20,"owner":"509:Ch-owner,0b-client,0-buyer,ledger-ST-North Carolina,C405:Ch-ca.org1.example.com,0-org1.example.com,L-Durham,ST-North Carolina,C405"}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -c mychannel -n private -c '{"function":"ReadAssetPrivateDetails","Args":["Org1MSPPrivateCollection","asset1"]}'
{"assetID":"asset1","appraisedValue":100}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_LOCALMSPID="Org2MSP"
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_ADDRESS=localhost:9051
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -c mychannel -n private -c '{"function":"ReadAsset","Args":["asset1"]}'
{"objectType":"asset","assetID":"asset1","color":"green","size":20,"owner":"509:Ch-owner,0b-client,0-buyer,ledger-ST-North Carolina,C405:Ch-ca.org1.example.com,0-org1.example.com,L-Durham,ST-North Carolina,C405"}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -> localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"ReadAssetPrivateDetails","Args":["Org2MSPPrivateCollection","asset1"]}'
Error: endorsement failure during query, response: status:500 message:"failed to read asset details: GET_STATE failed: transaction ID: 047d84d4a1a30e7119296e1deb756a807468f5572a7f61a3d28ab86: tx creator does not have read access permission on privatizedata in chaincodeName:private collectionName: Org2MSPPrivateCollection"
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export ASSET_VALUE=${echo -n '{"assetID":"asset1","appraisedValue":100}' | base64 | tr -d '\n'}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode invoke -n localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"AgreeToTransfer","Args":[""]}' --transient '{"asset:assetID":"$ASSET_VALUE"}'
2022-07-24 21:46:53.047 IST 0001 INFO [chaincodeCmd] chaincodeInvokeStubQuery -> Chaincode invoke successful, result: status:200
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -c mychannel -n private -c '{"function":"ReadAssetPrivateDetails","Args":["Org2MSPPrivateCollection","asset1"]}'
{"assetID":"asset1","appraisedValue":100}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_LOCALMSPID="Org3MSP"
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_ADDRESS=localhost:7051
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_LOCALMSPID="Org3MSP"
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/org3.example.com/users/owner.org3.example.com/msp
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export CORE_PEER_ADDRESS=localhost:7051
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -> localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"ReadTransferAgreement","Args":["asset1"]}'
{"assetID":"asset1","buyerID":"509:Ch-buyer,0b-client,0-buyer,ledger-ST-North Carolina,C405:Ch-ca.org2.example.com,0-org2.example.com,L-Hursley,ST-Ramphire,C405"}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 export ASSET_OWNER=${echo -n '{"assetID":"asset1","assetOwner":"Org3MSP"}' | base64 | tr -d '\n'}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode invoke -n localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"TransferAsset","Args":[""]}' --transient '{"asset:owner":"$ASSET_OWNER"}' --peerAddresses localhost:7051 --tlsRootCertFiles "$PWD/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt"
2022-07-24 21:47:22.800 IST 0001 INFO [chaincodeCmd] chaincodeInvokeStubQuery -> Chaincode invoke successful, result: status:200
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -> localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"ReadAsset","Args":["asset1"]}'
{"objectType":"asset","assetID":"asset1","color":"green","size":20,"owner":"509:Ch-buyer,0b-client,0-buyer,ledger-ST-North Carolina,C405:Ch-ca.org2.example.com,0-org2.example.com,L-Hursley,ST-Ramphire,C405"}
chaicha-8450M-8000S-5111E::채팅회원 /Andes/Andes/Andes, fabric/test-network5 peer chaincode query -> localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$PWD/organizations/ordererOrganizations/example.com/orders/orderer.example.com/msp/tlscaerts/tlsca.example.com-cert.pem" -c mychannel -n private -c '{"function":"ReadAssetPrivateDetails","Args":["Org3MSPPrivateCollection","asset1"]}'

```

[그림] asset 거래

위 캡처에서 org1과 org2의 asset을 교환하는 것을 볼 수 있다. asset2가 buyer고 asset1이 onwer인데 buyer가 자산을 받는 것을 허용하면 asset1에서 자산을 그만큼 없애고 asset2에서 자산을 추가한 것을 볼 수 있다. 차후에 data structure를 수정한 후에 private data를 적용해서 자산 거래가 가능하도록 만들 예정이다.

2) Node.js 서버 및 html(javascript)



```
JS index.js U x  homepage.html U  addition1.html U  addtion2.html U
html > JS index.js > ...
1  const express = require('express')
2  const shell = require('shelljs');
3  const app = express()
4  const cors = require('cors');
5  const port = 3000;
6
7  app.use(express.static('public'));
8  app.use(cors());
9
10 app.get('/', (req, res) => res.send('Hello World!'))
11
12 app.get('/homepage', function(req,res){
13   var command = 'peer chaincode query -C mychannel -n ledger -c \'{"Args":["GetAllAssets"]}\'';
14   var result = shell.exec(command);
15   result = result.slice(1, -2);
16   res.send(result);
17 })
18
19 app.get('/finedust', function (req, res) {
20
21
22 })
23 app.get('/forecast', function(req,res) {
24
25 })
26
27
28 app.listen(port, () => console.log(`Example app listening at http://localhost:${port}`))
29
```

[그림] nodejs 서버 코드

nodejs서버를 통해서 html(javascript)에서 요청이 오면 터미널에 명령어를 실행하여 블록체인 네트워크에 접근이 가능하도록 설계했다. command는 string으로 이루어져 있어서 자유롭게 수정이 가능하다. 그리고 요청에 대해서 json형식의 데이터를 반환한다.

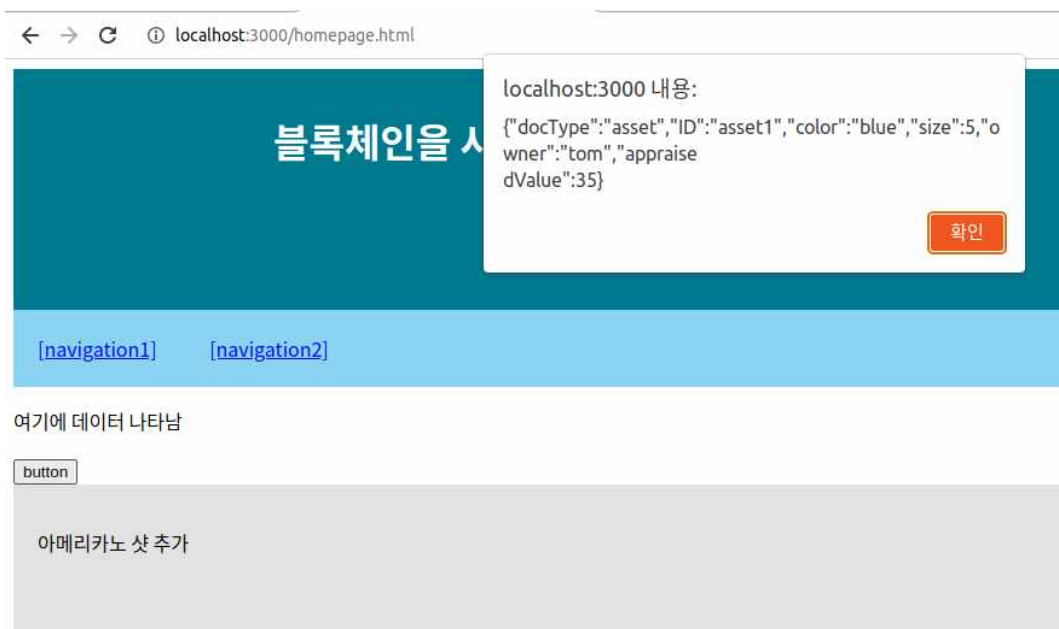
```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
    var xhr = new XMLHttpRequest();
    xhr.withCredentials = true;

    xhr.addEventListener("readystatechange", function() {
        if(this.readyState === 4) {
            var tbl = document.getElementById('json1');
            alert(this.responseText);
            data = JSON.parse(this.responseText);
            tbl.innerText = data.ID;
        }
    });
    function displayResponse(){
        var url = "http://localhost:3000/homepage"
        xhr.open("GET", url);
        xhr.send();
    }
</script>

```

[그림] javascript 코드



[그림] html에서 json 데이터 수신

alert를 통해서 json타입의 데이터를 정상적으로 수신하고 있는 것을 알 수 있다.



[그림]json 데이터 활용

그리고 버튼 위를 보면 json data를 활용하는 것을 볼 수 있다. (json data에서 asset2 데이터만 가져옴)